**Evaluación: JAVA**

Desarrolle una aplicación que exponga una API RESTful de creación de usuarios.

Todos los endpoints deben aceptar y retornar solamente JSON, inclusive al para los mensajes de error.

Todos los mensajes deben seguir el formato:

{​"mensaje"​: ​"mensaje de error"​}

# Registro

● Ese endpoint deberá recibir un usuario con los campos "nombre", "correo", "contraseña", más un listado de objetos "teléfono", respetando el siguiente formato:

{

​"name"​: ​"Juan Rodriguez"​,

​"email"​: ​"​juan@rodriguez.org"​ ,​

​"password"​: ​"hunter2"​,

​"phones"​: [

{

​"number"​: ​"1234567"​,

​"citycode"​: ​"1"​,

​"contrycode"​: ​"57"

}

]

}

* Responder el código de status HTTP adecuado
* En caso de éxito, retorne el usuario y los siguientes campos:

○ id:​ id del usuario (puede ser lo que se genera por el banco de datos, pero sería más deseable un UUID)

○ created:​ fecha de creación del usuario

○ modified:​ fecha de la última actualización de usuario

○ last\_login:​ del último ingreso (en caso de nuevo usuario, va a coincidir con la fecha de creación)

○ token:​ token de acceso de la API (puede ser UUID o JWT) (no es necesario)

○ isactive: Indica si el usuario sigue habilitado dentro del sistema.

* Si caso el correo conste en la base de datos, deberá retornar un error "El correo ya registrado".
* El correo debe seguir una expresión regular para validar que formato sea el correcto. (aaaaaaa@dominio.cl)
* La clave debe seguir una expresión regular para validar que formato sea el correcto. (Una Mayuscula, letras minúsculas, y dos numeros)
* El token deberá ser persistido junto con el usuario

# Requisitos

* Banco de datos en memoria, como HSQLDB o H2.
* Proceso de build via Gradle.
* Persistencia con Hibernate.
* Framework Spring.
* Servidor Tomcat o Jetty Embedded
* Java 8+
* Entrega en un repositorio público (github o bitbucket) con el código fuente y script de creación de BD.
* Entrega diagrama de la solución.

# Requisitos deseables

* Pruebas de unidad